AFRL-IF-RS-TR-1998-223
Final Technical Report
December 1998

MASS STORAGE PROTOTYPE

GTE Government Systems Corporation

Wayne J. Riesig and Sandra Fralick

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1998-223 has been reviewed and is approved for publication.

APPROVED:

PETER J. COSTIANES
Project Engineer

FOR THE DIRECTOR:

JOSEPH CAMERA, Deputy Chief
Information & Intelligence Exploitation Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFED, 32 Brooks Road, Rome, NY 13441-4114. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1998	Final Sep 95 - Sep 98

4. TITLE AND SUBTITLE
MASS STORAGE PROTOTYPE

5. FUNDING NUMBERS
C - F30602-95-C-0144
PE - 63726F
PR - 3192
TA - 30
WU - 09

6. AUTHOR(S)
Wayne J. Riesig and Sandra Fralick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
GTE Government Systems Corporation
Intelligence Systems Organization (ISO)
112 Lakeview Canyon Road, Box 5027
Thousand Oaks CA 91362-5027

8. PERFORMING ORGANIZATION REPORT NUMBER
1945062A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFED
32 Brooks Road
Rome NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-1998-223

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: Peter J. Costianes/IFED/(315) 330-4030

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The concept of accessing multiple repositories and the ability to treat those heterogeneous repositories as a single "virtual database" is a timely and worthwhile pursuit for the intelligence community. MSP provides the ability to create a "virtual database" from many legacy intelligence systems and allows for real-time correlative queries and union views of system related information at one time. This effort has succeeded in providing a tool to query the Imagery Exploitation Support System (IESS), Modernized Integrated Data Base (MIDB), and the Image Product Archive/Image Product Library (IPA/IPL) and returning a unified set of query results of information from each of thee databases. The MSP implementation of an intuitive web interface also provides a wide range of user's access to this federated view with less training than required of a full functionality client interface. The main COTS products used for this implementation were the Virtual DB (formerly marketed as PANGEA) by Enterworks, a subsidiary of TELOS and a Netscape browser. The MSP is also capable of retrieving thumbnails and full frame imagery from the IPA/IPL. Specialized queries for geographic processing and custom queries for co-located targets are also provided.

14. SUBJECT TERMS
Virtual Database, Heterogeneous Database Retrieval, Data Mining

15. NUMBER OF PAGES
44

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

TABLE OF CONTENTS

1.0 EXECUTIVE SUMMARY

The successful development and demonstration of the Mass Storage Prototype (MSP) is proof that there are currently commercial middleware products available to enable a federated view of multiple databases. As an Advanced Technology Demonstration (ATD), the program objectives were to develop a data model involving various intelligence databases that could be accessed with a single query. A primary objective was to implement this single query access using a commercial middleware product.

This ATD has shown that the technology currently exists to achieve this goal. The study phase of the MSP project was conducted during the time frame of October 1995 through March 1996, and at that time two middleware products were fully evaluated for the final choice. Since that time, many more middleware products have become available to access multiple databases. The technology has matured as has the product chosen for the MSP implementation, Virtual DB (VDB) by enterWorks.com.

The concept of accessing multiple repositories and the ability to treat those heterogeneous repositories as a single 'virtual database' is a timely and worthwhile pursuit for the intelligence community. MSP provides the ability to create a 'virtual database' from many legacy intelligence systems and allows for real-time correlative queries and union views of multiple system related information at one time. Basically, the intelligence community achieves more value from these existing systems over and above what they already provide while providing a single point of access for the user community. The MSP implementation of an intuitive web interface also provides a wide range of user's access to this federated view with less training than required of a full functionality client interface. This concept provides greater utilization at less cost per user.

2.0 INTRODUCTION AND SYSTEM CONCEPT

2.1 Background

The MSP program is an advanced technology demonstration program sponsored by the 480th Intelligence Group and managed by the Air Force Research Laboratory (AFRL) in Rome, New York. The objective of the program is to "deliver data management software that is applicable for multiple users, software applications, data bases and computer systems that are interconnected by a network and have access to mass storage systems."

GTE proposed to deliver a system based on commercial "middleware" that would provide MSP users with a single "Federated View" of the multiple intelligence databases maintained by the 480th IG. Such a system would maintain a "metacatalog" that defines the federated view, as well as mappings to the databases, tables, and fields that contain the actual data.

This system was then augmented to retrieve thumbnails and full frame imagery making it more useful as a user tool. Specialized queries for geographic processing and custom queries for co-located targets were also provided to satisfy user requests.

The MSP sponsor also requested a separate task for development and analysis of a network model. This model would provide a view into the network effects of MSP and allow 'what if' analysis on the location of the MSP server in relation to its target repositories.

2.2 Overview of Program

The MSP program is divided into four phases. The first phase is a study phase that defines the MSP system and identifies the commercial middleware product. The following three phases are cyclic development phases identified as Prototype I, II, and III. Each of these phases adds a repository system for query and includes user comments from the previous phase. Hence, the cyclic development allows each prototype to build on the previous one.

The Study Phase was conducted at the GTE facility in Rockville, Maryland, with additional engineering support from GTE in Thousand Oaks, California. The Study Phase identifies the overall system architecture and defines the criteria for the commercial middleware selection. It was during this phase that the web-based architecture was chosen. The middleware products were narrowed to two choices during the study: Interviso, by Data Integration Incorporated; and Virtual DB (VDB) by enterWorks.com, a subsidiary of TELOS (formerly marketed as Pangaea, by TELOS). After the selection process identified these two choices, it was decided to make the final choice during the Prototype I development cycle. This permitted analysis of the repository systems identified, in order to determine if they presented any specific technical challenges that would make one middleware product more viable than the other.

Prototype I continued with a technical analysis of the repository systems identified for access, which did identify a technical challenge of a requirement for an outer join capability. Interviso did not provide this capability and thus Virtual DB became the middleware product used by MSP. Prototype I development now continued on the solid base of a web based architecture with

---

[1] *Statement of Work for Mass Storage Prototype* PR NO. I-5-4720 3 March 1995

Virtual DB middleware. The object model was also established during this initial development cycle as a target and imagery based model with supporting information queried on related intelligence problems, requirements, reports, units and equipment. Completion of the Prototype I product included a working demonstration accessing two repositories: the Image Product Archive (IPA) and Imagery Exploitation Support System (IESS) databases.

Prototype II began with a change in team as well as venue. The development effort was moved to the GTE facility in Rome, New York with a final destination of the IE2000 Lab located at the Air Force Research Laboratory. This venue allowed for connectivity to actual IESS and IPA repositories. Prototype II began with integration with these live systems and continued with integration of the third repository, Modernized Integrated Database (MIDB), also accessible on the IE2000 network. During Prototype II geographic queries were implemented which accessed the IESS geographic search engine and then selected information on the targets found from all three repository systems. Retrieval of actual IPA imagery for display was also implemented during this phase.

During Prototype II, it was requested by the 480 IG that an early delivery be made to Langley Air Force Base (AFB) to allow users to begin their evaluation in the second rather than third prototype. The development effort was ahead of schedule at this time and allowed for this extra effort. A second server was procured by the 480 IG and delivered to the MSP team at AFRL for installation of the MSP system, including the Virtual DB product. This server was then returned to 480 IG at Langley AFB, and the MSP team integrated this early delivery on-site and provided training to 480 IG MSP users. The utility of this early delivery of MSP along with the time spent on-site by MSP Engineers proved invaluable in the definition of a working user interface that was co-developed by GTE and the 480 IG Intelligence Analysts.

Prototype II completed on schedule and under budget with a technical review and an MSP demonstration that was well attended by both 480 IG and representatives from USACOM as well. The USACOM representatives were so impressed by the MSP functionality demonstrated, that MSP representatives were invited to two separate briefings to discuss their potential use of MSP at USACOM.

During the technical review at the closure of Prototype II, the Demand Driven Direct Digital Dissemination (5D) system was chosen as the fourth database for access. Prototype III implemented both queries and imagery retrieval to the 5D repository. Extensive time was spent with the 480 IG users during this phase to ascertain their future needs from the MSP system and to understand their evaluation of the current capabilities. From these sessions, various tailored correlative queries were implemented to meet the needs of their daily duties. Prototype III also included integration upgrades to IPA/Image Product Library (IPL), IESS and MIDB systems. The IPL and IESS upgrades were virtually seamless as our metacatalog abstraction implemented within Virtual DB protects us from changes affecting our implementation. The MIDB upgrade was a major database schema change and did require re-engineering of the MIDB portion of the metacatalog to support this new schema. Due to the generic nature of our CGI-Bin code, no C code changes were required to support this new MIDB schema.

Prototype III completed with a technical review, on schedule and under budget, and included a successful demonstration of the expanded MSP functionality.

2.3 Overview of System Design

The MSP system design follows standard web site design conventions. The components of this design are the web browsers (e.g. Netscape, Mosaic) which reside on the client workstation, and the HyperText Transfer Protocol daemon (HTTPd) which services requests placed by the users and runs on the MSP server. The MSP software that services the requests of the users, placed through forms on the web browser, resides in an area known to the HTTPd server as the CGI-Bin (Common Gateway Interface Binaries). The MSP forms are written in the HyperText Markup language (HTML). The HTML code that makes up the forms is constructed dynamically by software written in the C programming language that resides in the CGI-Bin. The MSP system also uses a commercial middleware product (Virtual DB) that allows MSP the ability to provide the user with a unified view of multiple data repository systems through its metacatalog feature. The middleware's Application Program Interface (API) is used by the MSP CGI-Bin software to manage the query construction/execution and the data retrieval process. Brief descriptions of the major MSP components are as follows:

HTTP Daemon - The HTTP Daemon provides the interface between user requests, made from their web browser, and the CGI-Bin.

CGI-Bin - The CGI-Bin provides the forms that make up the user interface, builds and submits necessary query information to the Virtual DB API so it can construct Structured Query Language (SQL) to perform the searches, and provides a means to allow the users to retrieve thumbnails and images from IPA and 5D. All of the CGI-Bin routines are written in the C language and UNIX scripts.

Metacatalog - The Metacatalog allows the COTS (Commercial Off-The-Shelf) Virtual DB product to map the MSP logical data model to the physical database schemas. MSP created a three-tiered metacatalog in which the first tier mapped directly to the repository systems database tables, the second tier created the table joins and any data manipulation needed by MSP, and the third tier created the union view of all the repository systems. The metacatalog is created using the Virtual DB product and stored in the Gemstone Object Server, a component of the Virtual DB COTS package.

Virtual DB API - The Virtual DB API provides an interface for the CGI-Bin software to communicate with the Virtual DB metacatalog through the Virtual DB's object and data servers. It is part of the COTS Virtual DB package and consists of several software libraries that are used to create queries, return and format query results, as well as perform other administrative functions. The Virtual DB API provides a single point of access to manipulate all data returned from the disparate data sources using single routines versus having to handle results from each data source in it's native format in order to provide a federated data view.

3.0 DESIGN DETAILS AND ASSUMPTIONS

The purpose of this section is to describe, in detail, the assumptions and requirements for the design of this ATD for the 480 IG. The VDB product and its related components also will be discussed.

3.1 Introduction and Assumptions

MSP was built in four phases. In the study phase the commercial product (Virtual DB by enterworks.com) was selected and a preliminary design was developed. A cyclic design process was employed which allowed for user input at the end of each development phase. This cyclic development produced three prototypes. Each prototype built on an evaluation of the previous prototype and the addition of new functionality and a new database system with each delivery. The second and third prototypes were installed at the 480th IG, and were verified against the operational systems.

The MSP effort had three main technical thrusts:

- Providing a single point of access to multiple heterogeneous database systems
- Monitor and model the 480th IG network
- Retrieval of Imagery.

The goal was to develop a single query-only system that would read the data managed by four different intelligence systems. These "repository systems" are:

- Imagery Exploitation Support System (IESS)
- Image Product Library (IPL)
- Modernized Integrated Data Base (MIDB)
- Demand Driven Direct Digital Dissemination (5D).

The system would interpret the query and use the metadata catalog to translate it into queries that were appropriate for each of the repository systems. It would then merge the results and present the user with a single display of data. The decision was made to use a "light client" interface for MSP. The MSP is implemented as a CGI-bin program running on a dedicated server and accessed via an http daemon. This caused a great deal of confusion within the intelligence community as other organizations were also developing HTML query servers. In the case of MSP, the use of HTML was only incidental to the main effort.

The principal goal of the data access effort was to perform the entire process with commercial products. All the retrieval logic was to be derived by the middleware based on the metacatalog only. No special-purpose retrieval routines were to be used. The systems being accessed are constantly undergoing revisions as new functionality is added. It was important to centralize MSP maintenance at the metacatalog, rather than having to re-code schema-specific retrieval routines.

3.2 VDB

VDB is comprised of several 'modules.' Three modules: the GemStone object server, the OMNI data server, and the Virtual DB C Interface (VCI) application interface are used during both the development and operation of the MSP system, while the Geode interface is used primarily during the development and test process.

3.2.1 GemStone - The Object Server and Metacatalog Manager

The GemStone object server is the VDB component that manages the metacatalog that comprises the virtual database for an application. It is responsible for maintaining all of the attributes of the repository systems and for translating the queries against the virtual views down to the base layer mappings and vice versa.

3.2.2 Geode - The VDB User Interface

The Geode interface includes several functions including, but not limited to, data server creation and maintenance, metacatalog creation and maintenance, and an Ad-hoc query tool used during development and integration testing.

3.2.3 OMNIConnect - The Data Server

OMNIConnect manages all socket connection software and handles conversion of requests to each database's native format.

VDB accesses the repository data sources through a Data Server. The VDB Data Server used by MSP is OMNIConnect. All connections, requests and operations performed on the repository data sources are managed by the data server. The data server also provides a command line interface that allows direct SQL to be executed against it. This interface can be called directly in the CGI-Bin code as a system call, or used by developers during integration testing.

The Data Server is responsible for performing the following tasks:

- Retrieving and storing schema of the repository systems
- Defining and storing any desired security controls used when accessing connected databases
- Accepting data query requests from VDB's object server
- Optimizing queries
- Communicating queries to connected databases
- Performing joins on data retrieved from connected databases
- Returning result sets to the VDB's object server.

OmniCONNECT supports several classes of servers natively, and can be configured to connect to them without any additional components.

Server Classes Supported (for OmniCONNECT):

- Access server
- DB2
- File server
- Generic
- Informix
- Ingres
- Local
- Oracle
- SQL server.

Databases that are not supported natively by OmniCONNECT can be connected using custom developed libraries. The following is a table of some additional databases OmniCONNECT can connect to and what is required for each.

Texis	Open Bridge for Texis
EDA/SQL	Open Bridge for EDA/SQL
Sherpa	Open Bridge for Sherpa

3.2.4 VCI - The C API

The VDB C Interface (VCI) provides a well-defined, controlled application interface to VDB. It shields the application developer from needing to know the internal detailed workings of VDB. It provides access to the functionality necessary for applications to interact with VDB. The VCI API allows front-end applications to be developed using any language, Graphic User Interface (GUI), or development tool which supports the use of a C library. The VCI functionality can be grouped into several logical groups.

Initialization and Configuration. Certain VCI structures and default configuration information must be initialized prior to use. This set of functions allows the developer to initialize the data structures and set and tune the configuration parameters to suit the needs of the application.

User Session Control. Any interaction with VDB must occur through a user session. A user session is established after a successful login to VDB has been accomplished. A set of VCI functions is provided to support numerous login, logoff, and other session-handling operations.

Metacatalog Access. All the views and attributes available in the virtual database are modeled and stored within VDB's Metacatalog. The VCI supports the traversal and retrieval of the Metacatalog attributes, also including information about the logical to physical data source mapping.

Query Maintenance and Execution. The VCI provides a set of functions to construct, save, and execute queries against the virtual database. There are also functions that allow the developer to access and query against the data repository systems directly - without going through the VDB translation layer.

Query Results Handling. The VCI enables the extraction of query results and provides utilities for presenting the results in either a tabular format or a custom format built by the application by creating a data retrieval object. The actual SQL statement that is sent to the repository systems can also be requested through VCI functions.

Error Handling. All VCI functions return a status code that indicates the success or failure of the operation. Upon an unsuccessful operation a negative value will be returned. This value can then be sent to other VCI functions for decoding so that a text error message can be received.

3.3 Data Model Design

The MSP data model was the basis for the proof of concept for this ATD. This data model was driven by the fact that we were only given permission to access databases contained on the 480 IG intranet. After evaluation of the database repositories available on this intranet, it was determined that a target and imagery based model could best exercise the single query technology and allow us to prove all aspects of this ATD.

The MSP data model supports target and imagery queries along with related queries for reports, requirements, imagery, units and equipment. This object model is accessed by a series of queries defined as primary and related searches.

The primary searches are against the Target, Image, Intelligence Problem, Unit, and Equipment objects. A primary query is supported by an HTML screen that contains the attributes available for query on that object. Once the user enters the query data, a primary results screen is displayed with the hits for that single query, along with identification on which repository system from which the hit was retrieved. The user can click on each primary result row for a display of more detailed information on that hit row.

If further information is desired, a hit row can be chosen and then related queries specified on that record. The user need not enter any data other than a date range of interest to generate the related queries. The required data to build a related query is retrieved from the details of the hit record chosen and the query is automatically built and submitted to the multiple repository systems.

3.4 Metacatalog Design

VDB provides for a two-tiered approach in relation to implementing the metacatalog to support the object model being defined. The two tiers are defined as the default views and the user views.

The default views are created when table definitions retrieved from the repository system schema are loaded into VDB. This loading is handled by the OMNI defgen process. First, a database is created which points to the physical system and then tables within that database are specified for loading. All tables can be loaded or specific tables chosen. MSP has loaded only those tables required to support the defined object model. This means that changes to tables in the target database, not accessed by MSP, are of no concern. There may be 20 tables in a target system and possibly only 2 or 3 are necessary to support the defined object model. In this case, it is not necessary to carry the overhead of concern for the entire system. When the specified tables are loaded, the default views are automatically created in the metacatalog. These default views consist of each loaded table definition including its associated attributes that appear exactly as they were created in the repository system

The second level created within the MSP metacatalog architecture is referred to as 'database views.' This second level of abstraction, used in the MSP metacatalog implementation, is defined using virtual views based on the default views/tables that have been loaded. These views support the data objects defined in the MSP object model. The database view abstraction provides three major functions:

1. Projection of only those attributes that are of interest for query and/or display.
2. Definition of foreign keys to join multiple tables required to support a single data object.
3. Data manipulations required on base attribute datatypes.

Within these virtual views, only those attributes of interest from the default views will be created. This once again lets MSP further isolate itself from changes to the target repository attributes that do not affect the data model.

In our creation of a database view to support the image object, for example, we may include attributes that are contained in multiple default views or physical tables. Therefore, the default level joins must be defined. These joins are defined by created foreign keys that relate the tables together. These foreign keys can be defined as equi-joins, inner or outer joins, etc. Once a foreign key is defined between two tables, that join will automatically be included when fields are chosen in a single query from those tables without any further intervention.

Any data manipulations required to specific attributes are also defined at this level of the metacatalog. These data definitions could be required to support union view definitions that require a uniform attribute definition across the members of the union. Data manipulations can also be required to display data in a more usable format, etc.

The third level defined in the MSP metacatalog is referred to as the 'user views.' This third level of abstraction is defined as union views. These union views are what the MSP CGI-Bin accesses to support the user queries. The union views define the projected attributes for user selection and query. Once these attributes are defined, the database views created at the second catalog level are used to project these unions. For example, the Target union for MSP contains projections from the database levels views for IESS, IPA, MIDB and 5D. This third level of abstraction is the only level of the metacatalog a programmer is aware of in querying all four repository systems. These are also the views a user would be given access to, if ad hoc queries were allowed.

3.5 User Interface Design

The MSP user interface was developed entirely in the 'C' language with the addition of several UNIX shell scripts. The 'C' code makes up the MSP CGI-Bin and creates HTML web pages that are sent back and forth to the user through an HTTPd.

The user interface was developed with the following goals in mind:

- Generic design that would accommodate easy modification
- Layered in such a manner as to ensure that most changes are made within the 'high-level' code and not down at the 'utility' level
- Flexible so as to accommodate additional requirements throughout the program life cycle
- Standard HTML look and feel.

These goals were defined in direct support of the operational system administrator. Any viable intelligence system must be designed such that it can be maintained quickly and efficiently by the on-site system administrators.

The MSP user interface consists of the following parts:

Search Forms. The search forms are the only part of the MSP system in which the users have direct contact. All other functions are hidden within the rest of the user interface or the Virtual DB product. There is a search form, for each class of search, which the user must fill out prior to submitting a query for information. All of the forms have a standard look and feel and operate in the same manner regardless of the search being performed.

HTML Parse Functions. Each time a form is submitted to MSP, or the results of a query are returned to the user, the data must pass through the daemon and be parsed prior to use. MSP provides a set of functions that handle the parsing when data is received, and also the encoding of data leaving. These functions conform to the HTML 3.0 standards.

Software Data Bus/Query Parse Functions. Once the data has been parsed in to name/value pairs it must be sorted and arranged in a manner so it can be used to construct a query, if necessary, or perform other necessary operations. Due to the early requirements, that were eventually lifted but much too far into the development cycle to help, the MSP development

team was limited to the HTML 2.0 command set. We therefore, decided to implement a data bus structure that would be sent back and forth to the user throughout their session and that this data bus would carry the information that was required, and be updated when necessary. The data bus is designed in such a manner that it has both variable and fixed fields for efficiency and is flexible enough to handle any 'special' case data required. The data bus is a four-part structure defined as follows:

1. This first part contains a fixed number of fields, that can be expanded if necessary, and which holds all of the users information required; such as: username, previous searches run, and other custom selections they may have made.

2. The second part, which is variable in length, holds all of the information required for the search to be performed; such as: search fields, the view against which the search would be performed, and the search criteria.

3. The third part holds the users data base selections.

4. The fourth part is variable in length and holds any miscellaneous information unique to the specific search or request being performed.

The set of functions described above, further parses the data and places it onto the 'bus' in the correct manner as required.

Query Building Functions. Once the user has filled in a search form this group of functions will begin to build the SQL statement that will query the metacatalog. These functions have the capability to automatically implement, based on the search form design, wildcard 'LIKE' qualifiers as well as negation, between, Or, and In operations.

Construction of a query that goes against the metacatalog can be accomplished in two primary ways. In both instances the query will reference attributes that are defined in the virtual database view of the metacatalog.

The first and most straightforward method is to build a literal string that contains the exact SQL statement that is to be executed. One drawback of this method is that the SQL is not checked for validity until it is executed.

The second and more flexible method is to build the SQL in three parts, by using the VCI's predefined data structures, and allow the VCI to create the literal SQL string. This allows the user to generate query definitions that are dynamic in nature based on the application users wants and needs. The developer also has the benefit of error checking that is provided by the VCI throughout the query construction process. The process is basically broken into the logical parts of a typical SQL statement. The developer must supply the SELECT list, FROM and WHERE clause attributes, and finally any grouping and ordering attributes. Throughout this process error checking is employed by the VCI and therefore the developer can attain a high degree of resolution with regard to exactly where in the SQL the errors, if any, have occurred. The errors reported by the first method, the literal SQL string, discussed above are very general in nature and therefore may require more time and effort to debug.

The hard coded, or literal, query strings can still be used for more fixed applications.

Any query definition that has been created by either of the above methods can be stored and later referenced if desired.

Virtual DB Interface. This function provides the singular interface to the Virtual DB VCI. It is responsible for logging in to VDB, completing and submitting the query, retrieving the results and logging out of VDB. It also provides for 'passthru queries' that go directly against the repository systems and bypass the metacatalog altogether.

Result Handling Functions. Once the results are returned from the query, selected fields are displayed in a custom order by this set of functions. Results are received from the VCI in a two step process. The first step is to return parameters about the results, e.g., the number, size, etc. This allows the CGI-bin to allocate enough memory to accept the results into a local data structure for future manipulation. Once the parameters are known, data structures are sized and the results are requested. When the results are received from the VCI, they will be delimited and will need to be 'split' into the fields that make up each result row.

4.0 IMPLEMENTATION AND INTEGRATION

The front-end application provides the interface between the user and the metacatalog. This allows the developer to present the user with a view of the many databases as a singular entity. It does not preclude the user from attaining access to just one of those databases, if required.

The front end of an application that employs the VDB product interfaces with an API referred to as the VCI (Virtual DB C Interface). It is through this interface that an application gains access to the metacatalog terms and query results. Because the metacatalog removes much of the maintenance of the metadata from the front-end application it becomes very easy to adapt the front end to changes within the metacatalog. If properly designed the front end will hold singular references to elements within the metacatalog at the virtual view level. These references will be used throughout the front end and only need to be modified, in a single location, if they are changed within the metacatalog. If the underlying, or default view, attributes structure or content is changed the reference in the front end does not need to be modified as it generally interacts only with the virtual view layer of the metacatalog.

Although the metacatalog provides a nice neat method of isolating the front end from the data repository systems it does not prohibit the developer from constructing queries that go against the underlying databases directly through the use of a pass-through query. MSP made use of the pass-through query on the initial implementation of geographic queries. This pass-through query method allowed MSP to directly invoke the IESS Geographic Search Engine, which is a Sybase Open Server Process. Either the pass-through query method or the stored procedure functions both allow for execution of stored procedures resident on the repository systems.

5.0 NETWORK MODELING

5.1 Overview and Objectives

The original plan for the Network Modeling effort was to produce a network model that could be used for multiple functions. The MSP network-modeling task had two objectives. Its first objective was to produce a generic model of a MSP System. The generic model would describe the interaction between the MSP Client Workstation, MSP Server, and the MSP Data Repository Systems (IESS, IPA/IPL, MIDB, and 5D).

The second objective was to produce a model that could be used to predict how MSP System would behave in the Air Force Intelligence Network (AFINTNET) environment at the 480 IG. An additional function of this model was to provide a tool that the 480 IG could use to evaluate changes to the AFINTNET topology

The communications infrastructure at the 480 IG is the Air Force Intelligence Network. AFINTNET consists of an Asynchronous Transfer Mode (ATM) backbone connected to a series of FDDI rings. The rings are organized on a Community of Interest (COI) basis, and support connections to the individual COI database systems. Each ring is also bridged to an ethernet, which supports the COI workstations. All workstations have connectivity to all COI servers.

The network modeling objective was to capture this configuration in a network modeling tool (CACI's Network II.5) using the topology data collected by the network management station (HP's Open View). This would be coupled with probes at the COI hubs to derive background network traffic data. The activity due to MSP would then be measured and used to estimate the impact of MSP operations on the overall network.

5.2 Generic MSP Network Model

The task of developing a generic MSP network model was, itself, divided into multiple tasks. The first task was to determine whether a relatively accurate and realistic model could be created.

5.2.1 MSP Client/Server Interface Problems

Using a World Wide Web browser at a user workstation, the user sees a simple, intuitive mechanism for accessing information stored on World Wide Web servers, throughout the world. The user moves the pointer over a HTML link, clicks a button, and the next HTML *document* is immediately displayed on his screen.

The simplicity of the client interface hides the complexity of the process that must take place to display a *document* on the screen. The complexity arises from the design of the Internet and the Internet protocols.

One problem with using a WWW interface, as the mechanism for interacting with the MSP System, is the heavy demand placed on the network by the HTTP. HTTP is a transaction-oriented protocol that consists of a single client *request* followed by a server *response.*

A second part of the problem is the design and layout of HTML *documents*. Each page, frame, or graphical element that is displayed requires a separate HTTP transaction. For each HTML element, the client browser needs to determine how to route packets to a Domain Name System (DNS) name server, as per below:

1. Query the name server for the address of the WWW server for the element.
2. Determine how to route packets to the WWW server.
3. Establish a Transmission Control Protocol (TCP) connection to the WWW server.
4. Send an HTTP request to the server for the element.
5. Wait for a response from the WWW server.
6. Terminate the TCP connection with the WWW server.

The amount of network activity involved in downloading each HTML element raised questions about the level of detail needed in the MSP network model.

1. Was it absolutely necessary to include each step needed to establish a connection between the MSP Client and MSP Server?
2. Could the Reverse Address Resolution Protocol (RARP) broadcasts, used to locate the DNS and WWW servers, be eliminated?
3. Could the DNS queries be eliminated?
4. Could the TCP connection establishment and termination procedure be simplified?

Some of the questions were answered by experiments with the Network II.5 modeling tool. While the tool could simulate broadcast events in an ethernet environment, the principal concern was client synchronization. Should all of the MSP Client systems on an ethernet segment become synchronized, the resulting collisions and exponential backoffs will give the illusion that the MSP Server is overloaded or is not responding quickly enough.

5.2.2 MSP Client Browser Problems

If all MSP Clients use NCSA Mosaic or other Web browsers that obtains each of the elements serially, the potential problems may not be that significant. However, this could be a significant problem if the MSP Clients use a Web browser, such as Netscape Navigator, that initiates connections in parallel, in an attempt to minimize the overall time needed to display a document. By default the Netscape Navigator is designed to initiate four connections in parallel. This significantly increases the risk of collisions on ethernet segments.

With a large number of active MSP Clients, the default behavior of the Netscape Navigator increases the risk, that the number of connection requests arriving at the MSP Server, exceeds the TCP queue limit (or maximum connections). Traditionally, the queue limit and maximum connections are limited to five for systems based on the Berkeley Software Distribution (BSD)

networking software. This effectively permits eight connection requests (SYN packets) to be held, waiting for the next *accept* call from a daemon. When this value is exceeded, the networking software discards arriving connection requests.

5.2.3 Generic MSP Model Simplification

After some experimentation, it became clear that while NETWORK II.5 provided a mechanism for modeling broadcast query packets, it was cumbersome to use. It was discovered that a "race" condition existed. This condition occurred when the responses to a broadcast were being queued by the receiving system. If a second broadcast was transmitted before all responses to the first broadcast had been de-queued and read, the response to the first broadcast would cause a completion of the second broadcast. The model can be configured to discard messages whenever there is a response in the queue, causing the responding system to retransmit an acknowledgment of the broadcast to the receiving system. As a result, a decision was made to eliminate the Reverse ARP broadcasts for routing information from the model.

A similar problem occurred with the handling of DNS queries and responses; however, in this case the problem was how to simulate the expiration of DNS data held in the *name resolution* cache at each workstation. As DNS queries are relatively infrequent, and many military systems continue to use the older *host table* mechanism for name resolution, it was felt that the DNS queries could be eliminated from the model.

The three-way handshake used by the Transmission Control Protocol (TCP) to establish a connection was another problem area. This asymmetry was at odds with the *action/response* paradigm of the NETWORK II.5 modeling software. To simplify the complexity of the model, the connection process was redefined to be a simple transaction involving single connection request packet and an ACK or NAK response packet.

The above addressed the MSP Client/Server interface problems identified in Section 5.2.1. The problem that remained to be solved was the behavior of the MSP Client, discussed in Section 5.2.2. Again, the problem was one of complexity. If Netscape Navigator were used as the basis for the MSP Client, a problem arose with how to conditionally initiate 4 concurrent connections to the MSP Server.

Although the NETWORK II.5 documentation indicated that this could be done, attempts to create a model with more than one concurrent active thread failed. With version 10.1 of NETWORK II.5, there was no real mechanism to identify which of the processes in the model was to receive a particular response. Although CACI, the developer of NETWORK II.5, released several new versions of NETWORK II.5 in an attempt to resolve the problem, the problem remained.

5.3 480 IG MSP Network Model

The second network-modeling objective was to develop a MSP Network Model, based on the AFINTNET topology at the 480 IG. In addition to network topology, the goal was to incorporate samples of *typical* network traffic into the model. This would allow the MSP Network Model to be used as a general tool for evaluating proposed changes to the network topology.

The following are examples of questions that the 480 IG would like to have had addressed by the Network Model:

- What is the impact on the AFINTNET network, if the intelligence production groups shift from a limited number of users accessing the imagery systems, with system specific software, to allowing all members of the groups to access information on the imagery systems through the MSP Server?
- What is the impact on the analysts and other primary users of an imagery system of additional queries generated by the MSP Server?
- Where should the MSP Server be placed in the AFINTNET backbone?
- Should the MSP Server be placed on the same subnet as its most frequently accessed imagery system?
- What would be the impact on the analysts and other users of a imagery system if the MSP Server was placed on the same subnet?
- Should the MSP Server use the Mass Storage Subsystem (MSS) for storage of query results?
- If the MSS is used as a mass storage subsystem for the MSP Server, what is the impact on AFINTNET?
- Is one MSP Server sufficient? Or, are multiple MSP Servers required to provide a reasonable level of service?

The NETWORK II.5 modeling software was selected, primarily, to address the 480 IG need for a tool to evaluate network topology changes. It allows the model to be divided into two distinct components: (1) the process that is being investigated and (2) the network environment in which the process runs.

This distinction allows the model to be developed in two stages. The Generic MSP Network Model, discussed in Section 5.2, addresses the process by which a user at an MSP Client Workstation accesses the MSP Server, queries the server for information about data stored in the MSP Data Repositories, and retrieves the data from the repositories.

The key point is that it is the behavior of the process, not the process itself that changes when the network topology changes. As a result of this distinction, NETWORK II.5 was designed to allow importing topological information and network traffic from Cabletron Spectrum, Sun NetManager, and HP OpenView.

To develop the 480 IG MSP Network Model, the plan was, first, to incorporate the network topology from HP OpenView, that the 480 IG used to manage the AFINTNET. The topological data would replace the simpler, manually created network topology used to develop the generic model that described the MSP process.

The second part of the plan was to incorporate *typical* network traffic into the model. For the model all that was needed was the source, destination, and the amount of data being transferred. However, it was critical that the same network management software, HP OpenView, be used to provide the topology data to ensure that the source and destination systems existed in the model.

To obtain the information and data needed for the 480 IG MSP Network Model required the purchase of additional hardware and software to allow HP OpenView to capture network traffic information. In late 1996, an agreement was reached where the 480 IG would purchase three FDDI SAS RMON probes to be used for capturing network traffic. GTE would purchase the additional HP OpenView software components needed to support the RMON probes.

Due to changes in the AFINTNET topology during the first quarter of 1997 to meet 480 IG mission requirements, the number of probes purchased was reduced to two. Both the hardware and software needed to capture network traffic arrived at the site in early June; however, installation was delayed until late November 1997.

There were several reasons for the delay. The basic HP OpenView network management software had problems, monitoring the AFINTNET ATM switches, that were not resolved until early November. The 480 IG was in the midst of a mission critical upgrade to AFINTNET, thus the priority to support the network modeling effort, was secondary.

As a result, it was not until early December that it was discovered that changes to the HP OpenView software made it impossible to generate the ASCII files containing the network topology and network traffic that could be reviewed by the ISSO. HP Open View has not as yet provided the ability to generate these files in ASCII format.

5.4 Network Modeling Results

The results of network modeling effort were twofold. The first objective was completed successfully. GTE did produce a generic model of the interaction between the MSP Client Workstation, Server, and Data Repositories. The model is reasonably accurate. For any given network topology, the performance predicted by the model is similar to what a user at the MSP Client Workstation would experience on a relatively low traffic network.

The modeling of the 480 IG AFINTNET environment was not as successful. Neither GTE nor the 480 IG were able to capture the network traffic information needed for the model, in a format that could be reviewed by the ISSO to ensure that classified information was not compromised.

The failure to collect the network traffic information was the result of software changes to the HP OpenView network monitoring software and changes to the AFINTNET topology needed to meet operational requirements of the 480 IG.

As a result of the topology changes, only one of the FDDI RMON probes acquired for data collection could be installed. In addition, changes to the Cabletron MMAC Plus firmware and software allowed the ethernet traffic to bypass the hub's FDDI backplane, thus eliminating most of the traffic that was expected to be captured for the network model.

Additionally, fatal to the data collection, were changes to the latest release of HP OpenView needed to monitor the AFINTNET ATM Backbone. While the new graphical interface allows the network manager to view network traffic as it is collected by the RMON probe, and to direct the raw binary data to a disk file for later review, using the same graphical interface, it no longer supported the generation of a flat, ASCII text file, that could be reviewed by the ISSO.

Work on the Network Modeling effort concluded as of 31 March 1998. At the April TEM, at the 480 IG, the network modeling status and results were briefed. The Government and GTE both agreed, that no additional effort should be expended on network modeling. However, it was requested that results and recommendations be included in the Final Report.

6.0 TECHNOLOGY ADVANTAGES AND DEFICIENCIES

The following paragraphs document the advantages and deficiencies of the current technology of accessing a federated view of multiple repository systems, as well as some of the specific advantages and deficiencies of our chosen COTS package, Virtual DB.

6.1 Technology Advantages

6.1.1 Producibility and Supportability

One of the benefits of this technology is the portability and ease of which the system can be developed and transferred or installed at different locations. This is largely due to the metadata isolation achieved within the metacatalog maintained by Virtual DB. The three level metacatalog abstraction, developed by MSP engineers, and the generic nature of the CGI-Bin code described above, allow for changes in the repository systems to be accounted for very rapidly and generally without any CGI-Bin changes required. Using a COTS product, such as Virtual DB, significantly minimizes the quantity of 'in-house' software needed to be developed and also reduces the long term maintenance required

A system, like MSP, developed with Virtual DB, can be transferred to another site that has the same repository systems and configuration, and will be virtually seamless. In an architecture such as that employed by MSP, there would be no required changes to the application front end or the metacatalog. The only changes that would be required are within the OMNI Connect database. The IP address of the repository systems, as well as their site names would need to be updated within the OMNI configuration to allow for communications. This is a two step process, accomplished through the use of a command line tool and a function within the VDB Graphic User Interface (GUI). In order to obtain the best performance, an OMNI 'update statistics' function should be run on these new repositories. This is an automated procedure that can be left to run unattended, and will generate a statistical spread analysis on the indices contained in the repositories, thus allowing for optimum query plan generation.

This benefit was realized significantly within the MSP development and deployment cycle. There were multiple version differences in all repository systems between the MSP development site and the 480 IG deployment site. In general, an MSP upgrade to the 480 IG site was installed and ready for user access in as short as one day.

6.1.2 Using Multiple Databases as One

The main benefit of creating a federated view in a COTS product, such as VDB. is the ability to treat these heterogeneous databases as if they were one database. For example, tables contained in two different databases/systems can be joined as if they were resident in the same repository. This allows for queries and functionality beyond the union concept, basically, providing the capability for data fusion.

This capability also aided MSP in working with the 480 IG IESS system, which is a two Sybase Server configuration. The two server configuration of IESS places the information, pertaining to reports and imagery, on one server, and the information on targets contained in the Exploitation database on another. Tables, contained within two different Sybase Servers, cannot be joined

together. In the IESS product, application code has been written to handle this situation. Using the VDB suite, MSP was able to directly join the Exploitation and IIR and Exploitation and Imagery_Coverage tables together, directly based on BENumber with no custom application code required. The processing of this cross server join is accomplished within the OMNI Connect product. Handling this cross server join within OMNI causes no extra load on the IESS Servers, as does the IESS application solution to this cross server join.

Within the MSP data model designed for the MSP demonstration, no situation existed to exercise the data fusion capability. This capability, which is supported by VDB, allows for major intelligence gathering in one query that would require an analyst to manually fuse two different sets of information together to obtain the same results.

6.1.3 Ease of Adding Additional Data for Retrieval

Another advantage of employing this technology is the flexibility and ease with which the developer of the front end user interface application can make modifications without concern for the underlying data repository systems. All of the data handling, database joins, unions, etc., are managed within the metacatalog. It is the metacatalog that makes visible, to the front-end application, a virtual view of all the repository systems without the need to know their physical names or locations. Using this virtual federated view, the addition of new information for retrieval is a simple task. This was repeatedly proven during the installation and training exercises. Upon user request of the need to retrieve additional data fields, MSP engineers or suitably trained 'blue suit' system administrators could implement this enhancement literally in minutes.

The following paragraphs describe the steps required by the VDB System Administrator to retrieve a new field/attribute from a table already accessed within the metacatalog (Case 1) and the addition of a new field from a table not previously loaded within the metacatalog (Case 2).

Case 1. Addition of an attribute, from a table already loaded, to the DB level virtual view and present it to the user in the MSP user view;

To accomplish this requires two steps and, depending on the implementation, possibly one additional step. Since the table already resides in the default view of the virtual DB, the desired attribute within the table simply needs to be made accessible to the DB level virtual view, so it can be returned to the application. This is accomplished within the VDB metacatalog management tool and is simply a matter of 'adding' the desired field available in the default view to the user view. Once this is done, the attribute will be projected in the MSP user view, a single step in the metadata catalog management tool, and returned whenever the user view is queried. An additional modification to the application, that of processing the results once they are returned from VDB, may be required depending on implementation. Since MSP implemented 'Select *' queries, VDB will always return all the attributes of the queried view. They will be returned to the application in the 'natural order' that existed within the view. MSP implemented a scheme by which the returned data set can be 'virtually' reordered at will, so presentation of the data to the user can be made more effective. The data that drives this feature is stored within a C header file. A slight

adjustment to the header file will need to be made if the addition of the new attribute disrupts the desired data reordering.

Case 2. Addition of an attribute, from a table not previously loaded, to the DB level virtual view and presents it to the user in the MSP user view.

To accomplish this requires one more step than case 1 above. Since the table does not currently reside with the virtual DB default layer, it will need to be created before any of its attributes can be referenced. This is a two step process. First, the table must be Def Gen'd into the OMNI database. This is the process of loading the new table definition and is an automated procedure that is part of the VDB tools. This automated procedure will make the physical definition of the table known to OMNI Connect, and automatically generate the default level metacatalog terms. The second step is to add the attribute to the database level views and add the foreign key definitions between the new table and any pre-existing tables. When these two steps have been completed, the steps outlined in case 1, above, will need to be followed to make the new attribute visible to the application, and thus the user.

As shown above, once the object model is defined within the VDB metacatalog, manipulations to that model are extremely easy to implement

6.1.4 Separate API's are not Required to Access Repository Systems

The ability to access repository systems, *without* use of a custom API on each repository, is a major benefit of VDB technology. The only requirement is a 'read only' account on the repository database systems. Three major benefits are realized: 1) No waiting for a requirement to be levied and satisfied on a legacy system to support an API; 2) Developers are not at the mercy of changes made to the API; and 3) Developers can tailor queries and returns to be exactly as required, and maintain control of enhancements to these queries.

In addition, another tangible benefit is that the development team will only need to interface with, learn, and be concerned with modifications to the Virtual DB C Interface. By using Virtual DB VCI the development team has complete control over all the data that would ordinarily be handled by each custom API. All queries and results are managed by this single API, the VDB VCI, rather than having to manage queries and returned results to and from several different systems in differing formats

6.1.5 Catalog Migration Feature

Virtual DB includes a migration process that allows the migration of all, or a subset of, the objects from a development Virtual DB metacatalog to a production Virtual DB metacatalog. This feature greatly reduces the time needed to upgrade and maintain production systems. Once the additions/modifications have been made and tested within the development environment the upgrade of the production system can be accomplished within a very short period of time. The on-site MSP engineer used this feature extensively at the 480th Intelligence Group and at the MSP development site with significant success. The production site never required more than 2-3 hours to effect each upgrade using the migration feature.

The migration process is basically a two step process. The first step requires specification of the objects that reside within the development system which will be migrated to the production system. This step creates a textual report that defines the objects specified for migration. Once the report has been verified it is used as input to create a binary migration data file that will be used to perform the migration on the production system.

Upon completion of the first step, the resultant data file is then taken to the production system and used as input to perform the actual migration. Once the migration load is completed, the production system will contain all of the objects that were specified for migration within the development system.

The following factors should be considered when making use of the server migration feature.

- The data model, on the production database, should not be "independently modified" apart from the development system.
- All subsequent creation and modification of objects within the production system should occur via server migration loads from the development database. In practical terms, the data models of well bounded, and yet coordinated applications, may have different origins. If this is the case, the onus is then on the data modeler and migration specialist to avoid both conflicts and false assumptions of sameness.
- During the migration process on the production system, each object to be migrated is examined, and it is determined whether it is an existing or new object. This decision is made based on the object's name. If it is an existing object, then that existing object will be updated. If it is a new object it will be created.
- If applications, developed separately, choose the same name for their objects, migrating them to the same target server will cause a merging. There may be cases where that is the desired effect, but the data modeling process must be aware of this possibility, and choose names accordingly.

6.2 VDB Specific Advantages

The following paragraphs highlight the flexibility and features provided by Virtual DB by enterWorks.com, the COTS product chosen by the MSP Program.

6.2.1 Platforms and Operating Systems

Virtual DB supports numerous platforms and operating systems for both clients and servers. Servers must operate on an X window system, and clients can operate on any platform that supports network protocols.

Server Platforms and Operating Systems

- Solaris v2.4, 2.5
- AIX v3.2.3, 4.1
- HP-UX v9.0.

Client Platforms and Operating Systems include any system supporting a web browser.

6.2.2 Access to Multiple Data Sources

Virtual DB allows applications to be developed that have the capability to access multiple data repository systems, simultaneously using a single query. Data integration involves accessing and combining data from multiple heterogeneous corporate data sources with location transparency. Connection management to the multiple sources, cross-database joins and unions, and query language translation are provided to allow users to query data without regard for its physical location or structure.

6.2.3 Access to Diverse Data Sources

Virtual DB provides support for numerous types of data repositories. If a special type of data repository is not supported, then an open access server is provided to integrate it using custom methods.

6.2.4 Access to Diverse Data Formats

Virtual DB does not constrain the object-oriented technology to certain defined data types, and is able to access any type of data format. Both structured and unstructured data formats are included in the Virtual DB. Business data is stored in either a structured or unstructured format. Structured data is stored as attributes in a relational, hierarchical, or network databases. Structured formats are fairly easy to integrate from multiple sources. Unstructured data sources consist of images, text databases, audio files, video files, and other types of information. Unstructured formats need to be incorporated to allow a complete information system. Virtual DB provides the flexibility to include unstructured data formats and stands apart in how it addresses these data sources.

6.2.5 Data Joins Across Multiple Sources

VDB provides the ability to join data from multiple sites recognizing that data resides in many different physical data repositories. The capability to join that data without coding the client application to retrieve multiple data sets and process the data locally is a major benefit. VDB not only communicates with each data repository in an optimized fashion; it completely shields client applications from the knowledge of where the physical data is stored.

6.2.6 Open Architecture

During data integration, there is usually a "special case" type of data. For some, the special case is an almost obsolete repository that is on "life support" from the original vendor. The system is very valuable and time does not permit an easy migration or replacement for the existing source that is going obsolete. It is necessary, that the existing source be used. This special case is normally the demise of most efforts to create a nice single interface into all corporate data. Virtual DB provides the answer of keeping the existing data by providing a very OPEN architecture. Virtual DB is able to incorporate "special case" data repositories into the data model, and the front end applications no longer need to have custom code to account for special case data. The existing data can be accessed from Virtual DB, and that the data can be cleansed (manipulated) to fuse with other retrieved information.

6.2.7 Isolation of Application

Virtual DB has isolated the application from any data logistic issues, that might have hindered application development, when trying to use other technologies. This has been achieved by:

- Ability to incorporate "NEW" data sources by writing an access server for them
- Ability to incorporate diverse data sources
- Ability to access proprietary flat file database systems
- Ability to access non-mainstream existing systems
- Ability to access prototype systems
- The object server consisting of an active Object Oriented Database Management System (OODBMS) and special objects that maintain metadata and business rules.

6.2.8 Data Migration

VDB provides migration from an existing data source over to a new data repository (e.g., Moving from VSAM on the main frame to Sybase on Solaris). In almost any corporation, there are some systems that have outlived their usefulness. The problem that plagues most businesses is that there have been multiple applications written to interact with those obsolete systems. In order to eliminate the old systems, the applications must be ported first. Virtual DB provides an interface that can integrate the old obsolete system, while applications are ported and written to target VDBs higher level corporate model of data. This will free applications from being coupled to a specific data repository in the future. The actual migration of data from the old systems into the new systems can then be automated over time, through VDB into the desired data repository.

6.2.9 Data Warehousing/Data Cleansing

Data Warehousing is the process of extracting, modeling, and storing data from operational systems and Data Cleansing is the process of removing erroneous and redundant data from a database. Data is often cleansed before mining, migration, or warehousing takes place. A common problem when merging data from disparate sources is that the data may be stored in different formats in the different data sources. In this case, it is desirable to standardize on one

scheme and "cleanse" the data from the various sources to have the same format. An example of this is the variety of date formats used by Database Management Systems (DBMSs). Assume the desire to standardize on a format of DD/MM/YYYY HH:MM:SS, with a 24-hour clock, for all dates in the system. However, the sample database is a Sybase database; its date format is MON DD YYYY HH:MM:SS AM (or PM).

This cleansed data can then be migrated from the operational systems into a data warehouse consisting of a totally new database schema. This new schema is defined in the VDB metacatalog and the existing data is extracted into this new structure and then actually transferred to a new database using the modified structure definition.

6.2.10 Ad Hoc Query Tool

The VDB product allows developers and users, on local or remote computers, to use either custom-developed client applications or the VDB Ad Hoc Query Tool, to access the information objects stored by VDB. The Ad Hoc Query Tool is invaluable during the development, integration and testing of an application.

6.2.11 Multi-Level Security

A Multi-Level Security scheme consists of one or more hierarchical levels, zero or more keywords, and a default specification. A security specification consists of a level and zero or more keywords. A user is allowed access to a security-controlled entity only if the user's security specification has a level that is the same as or higher than the level in the entity's security specification, and if the user's security specification includes every keyword in the entity's security specification. In other words, if the user's security access is equal to or higher than that of the entity, the user is allowed access. The site ISSO approved the concept employed by VDB to provide access if the users security access is higher than that of the entity contained in the default specification, the user is allowed access.

6.3 Technology Deficiencies

6.3.1 COTS Upgrades

Upgrades to "back end" repository systems might not be directly supported by the middleware product, or it may take some time for the middleware vendor to develop, test and deliver. The vendor may also not plan to provide the necessary support, if it is a special case, or not needed by other users of the middleware product. Therefore, there are some risks involved, by putting some of the software in the hands of a third party. However, this risk is mitigated by the benefits of not having to develop and support custom code to perform the many functions provided by the COTS products.

6.3.2 VDB Learning Curve

As with any new tool, there is a learning curve associated in its initial use. The VDB tool is currently undergoing development of a very different GUI, from that currently provided. The MSP engineer and 480 IG site system administrators have seen and understand the initial concept of this new interface and believe it will significantly ease the current learning curve.

6.3.3 Data Integrity Issues

This issue is documented here, ***although it is not an issue only with MSP or middleware technology***. In implementation of our data model to exercise this ATD, we encountered several instances of data integrity issues between the repository systems. For example, an OB TYPE on MIDB may be three characters in length, while the same field on IESS will only be two characters in length. This is an easier instance for solution, in that, one character can be inferred and translated on the IESS system. The harder instances to overcome are the entry of items; such as: unit name or equipment name, where no data integrity rules were applied on input and the conventions between the two systems are quite different. These types of data integrity issues make it nearly impossible to join between the two systems.

As previously mentioned, this is an issue that will be grappled with, no matter what the implementation, while trying to create a 'virtual database' from multiple databases. Greater emphasis and community enforcement of data integrity rules and data standards are sorely needed for this ongoing issue.

7.0 AVAILABLE TECHNOLOGY NOT DIRECTLY PROVEN BY MSP

The following paragraphs document some important features provided by VDB that were not able to be proven in this ATD, due to our limited data sources for access.

7.1 Accessing Diverse Data Sources

VDB provides support for many different types of data repositories. MSP, due to requirements and availability of data repository systems, only accessed Sybase databases.

The following is a list of databases that are directly supported by VDB:

- Oracle
- Sybase
- Ingres
- Informix
- TEXIS (also on NT Server)
- Flat files (including NFS mounted)
- IMS
- DB2
- IMDS
- ADABAS
- VSAM
- M204
- EDA SQL
- Sherpa.

VDB's open architecture is able to incorporate "special case" database repositories into the data model. Therefore the front-end applications no longer need to have custom code to account for special case data. The "special case" data can be accessed through VDB, like any of the other "standard" data types.

7.2 Accessing Diverse Data Formats

Accessing both structured and unstructured data formats is supported by VDB. Data can be stored in either a structured or unstructured format. Structured data is stored as attributes in relational, hierarchical, or network databases. Unstructured data sources consist of images, text databases, audio files, video files, and other types of information.

7.3 Data Migration

VDB provides migration from an existing data source over to a new data repository (e.g., Moving from VSAM on the main frame to Sybase on Solaris). In almost any corporation, there are some systems that have outlived their usefulness. The problem that plagues most businesses is that there have been multiple applications written to interact with those obsolete systems. In order to eliminate the old systems, the applications must be ported first. Virtual DB provides an interface that can integrate the old obsolete system, while applications are ported and written to target VDBs higher level corporate model of data. This will free applications from being coupled to a specific data repository in the future. The actual migration of data from the old systems into the new systems can then be automated over time, through VDB into the desired data repository.

7.4 Data Warehousing/Data Cleansing

Data Warehousing is the process of extracting, modeling, and storing data from operational systems and Data Cleansing is the process of removing erroneous and redundant data from a database. Data is often cleansed before mining, migration, or warehousing takes place. A common problem when merging data from disparate sources is that the data may be stored in different formats in the different data sources. In this case, it is desirable to standardize on one scheme and "cleanse" the data from the various sources to have the same format. An example of this is the variety of date formats used by DBMSs. Assume the desire to standardize on a format of DD/MM/YYYY HH:MM:SS, with a 24-hour clock, for all dates in the system. However, the sample database is a Sybase database; its date format is MON DD YYYY HH:MM:SS AM (or PM).

This cleansed data can then be migrated from the operational systems into a data warehouse consisting of a totally new database schema. This new schema is defined in the VDB metacatalog and the existing data is extracted into this new structure and then actually transferred to a new database using the modified structure definition.

7.5 Multi-Level Security

A Multi-Level Security scheme consists of one or more hierarchical levels, zero or more keywords, and a default specification. A security specification consists of a level and zero or more keywords. A user is allowed access to a security-controlled entity only if the user's security specification has a level that is the same as or higher than the level in the entity's security specification, and if the user's security specification includes every keyword in the entity's security specification. In other words, if the user's security access is equal to or higher than that of the entity, the user is allowed access.

8.0 MSP DOCUMENTATION

The following documentation was delivered as part of the MSP program. It can be referenced for more detailed explanations as to the analysis, design, and implementation of MSP.

- Interface Requirements Specification (CDRL A005)
- Software Design Document (CDRL A007)
- Software Programmers Manual (CDRL A011)
- Software Test Plan (CDRL A012)
- Software Test Report (CDRL A009)
- Software Users Manual (CDRL A010)
- System Segment Design Document (CDRL A004)
- Monthly Status Reports (CDRL A001).

9.0 RECOMMENDATIONS AND CONCLUSIONS

9.1 MSP Recommendations and Conclusions

It is our conclusion, resulting from the knowledge gained from the implementation of this ATD, that commercial middleware is a viable solution in providing a federated view of multiple repository systems. The MSP middleware product, Virtual DB by enterWorks.com, has allowed us to successfully and efficiently implement this ATD. The use of middleware allowed us to concentrate on our data model and the users needs rather than having to worry about each custom interface to the repository systems accessed. The scope of the efficiency of this method can be proven by examining the following:

1. A comparison of the original development schedule and initial requirements verses the speed of development and the extra functionality that was implemented on schedule and under budget.

2. The programs ability to implement an early delivery at the 480 IG, three months ahead of schedule.

3. The programs ability to successfully maintain that deployed site (480 IG) and the development site (AFRL), while the two sites were constantly out of synch in repository system versions.

4. The programs ability to maintain this extra deployed site (480 IG) while keeping on schedule with planned development.

As mentioned in this report, the middleware technology has matured greatly during the span of this ATD, and there are many more new products currently available, than at the start of the MSP project. MSP development, using Virtual DB, has proven successful, and the product should still be considered for future applications. However, it is further recommended that additional research of new products be made, prior to starting any new development programs.

During the development of the MSP ATD, two other programs were initiated: Sensor Box, by Air Force Intelligence Agency and Broadsword by Air Force Research Laboratory. MSP Engineers met with the Sensor Box representative near the end of the MSP development effort. Both MSP and Sensor Box were demonstrated during this meeting. Based on our short discussion and the Sensor Box demonstration, two entirely different sets of information were provided by Sensor Box and MSP. Both programs use Virtual DB as the middleware product; however, MSP is written using 'C' language (structured), while Sensor Box design makes use of 'Smalltalk' (unstructured).

Broadsword is similar to MSP in many ways, i.e., it does access repository databases (IESS and IPA/IPL), it does use HTML as a user interface, and there are some other similarities. However, the software development is strictly done by hard coding design. In comparison, MSP makes use of COTS products, e.g., Virtual DB as a middleware approach to accessing the databases. In our initial look at the Broadsword concept and its design approach, it was obvious to us, that the difference of not using COTS/Government Off-The-Shelf (GOTS) products and already existing technology (MSP) that needless money would be spent on that program. There had been some

discussions at Government meetings to look at incorporating MSP capabilities into Broadsword, i.e., the Gatekeeper function. At this time, we are unaware of those happenings.

Both MSP and Broadsword have been installed at 480 IG for analyst use. The 480 IG customer/ user has requested a comparison of Broadsword and MSP capabilities, as part of this report. However, MSP Engineering feels unqualified to perform this comparison without a full detailed demonstration of the current Broadsword capabilities. An MSP representative did attend a very short demonstration of Broadsword in June at 480 IG; however, due to a system problem a longer demonstration of the queries could not be shown. If this demonstration can be re-scheduled, or MSP Engineering could be given access to a Broadsword system, this requested comparison could be included in the final version of this report.

In regard to MSP, the product was integrated at the 480 IG, ahead of schedule, has been functional and available to users and analysts for their use, inputs, and recommendations. Many enhancements have been incorporated into the MSP product, as result of 480 IG comments and requests. It is the understanding of GTE, that the 480 IG is satisfied with the performance of the MSP Program.

9.2 Network Modeling Recommendations and Conclusions

At the MSP Technical Exchange Meeting held in February 1998 at the USAF Laboratory in Rome, New York, GTE was asked to address the following question in its final report. Should the Government include a standard, mandatory requirement for a network model in every Request for Proposal (RFP) involving software development? Answer: No, evaluate as an option only. Including a mandatory requirement for a network model on all software development projects may unnecessarily increase the cost of the development effort.

The Government should evaluate and assess the risks of the proposal, with regard to the software development methodology and software technologies used, before committing to the development of a network model. If the Government determines that the proposed approach only involves nominal risk, it need not exercise the network modeling option. For example:

Option: Not to Exercise

1. Our technical approach to MSP relied on using existing, WWW technology to minimize risk. This technology is understood and widely used by the Government. If the network model were a mandatory requirement, in this case, development of that model would have resulted in additional cost to the program.

2. MSP software development was based on using rapid-prototype methodologies. Typically, in this type of development, the modeling effort lags behind the software effort and, as a result, a network model may only be of limited value.

Option: To Exercise

1. If the technical approach involved developing a custom protocol for inter-process communication between the MSP Client Workstation and the MSP Server, there would clearly be risk involved. In this case the Government would want to exercise the option to ensure that potential problems could be identified early in the design and development cycle.

2. In actuality on the MSP Program, Network Modeling was used as a tool to evaluate network topology, monitor network traffic, and create "What Ifs," in order to obtain optimum performance of MSP and its association on the network, at the 480 IG. If the original AFINTNET configuration had not changed so drastically – the network modeling effort would have probably been a very good asset for the 480 IG.

10.0 SUMMARY

The main purpose of this report is to document the viability of developing a single query system using currently available COTS technology. The 480 IG was also provided with an operable intelligence tool as part of the MSP ATD. A brief discussion of that tool is warranted and described below

MSP provides 480 IG users with the following capabilities:

- Access to the MIDB, IESS, IPA/IPL and 5D repository systems in a single query.
- The MSP object model supports primary and related queries to the repository sites for target, image, report, unit, equipment, intelligence problems and requirement information.
- Geographic queries using Circle, Polygon, and LOC coordinates available for target queries.
- Thumbnails and images retrieved from IPA/IPL and 5D systems. Multiple thumbnails can be retrieved at one time.
- Abort functions provided to abort a database query or an image pull.
- Selective database query available to query only those systems necessary to the users and to handle cases where systems are unavailable.
- Ability to save query results in SDF files for import into reports or other applications.
- Specialized query was implemented to support user requests, which provides information on co-located targets appearing on a single full frame image.
- On-line Help.
- An interface has been provided for users to input problems and suggestions. These inputs are written to a file and are checked by engineers.

These functions are implemented using the web-based design described in the previous sections with Virtual DB as the middleware product. The 480 IG users and management have continually expressed their opinion that the MSP has been a useful tool.

11.0 NOTES

11.1 Acronyms and Abbreviations

AFB	Air Force Base
AFINTNET	Air Force Intelligence Network
AFRL	Air Force Research Laboratory
API	Application Program Interface
ATD	Advanced Technology Demonstration
ATM	Asynchronous Transfer Mode
BSD	Berkeley Software Distribution
COI	Community of Interest
COTS	Commercial Off-The-Shelf
DBMS	Database Management System
DNS	Domain Name System
5D	Demand Driven Direct Digital Dissemination
GOTS	Government Off-The-Shelf
GUI	Graphic User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IESS	Imagery Exploitation Support System
IPA	Image Product Archive
IPL	Image Product Library
MIDB	Modernized Integrated Database
MSP	Mass Storage Prototype
MSS	Mass Storage System
OODBMS	Object-Oriented Database Management System
RARP	Reverse Address Resolution Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
VCI	Virtual DB C Interface
VDB/Virtual DB	Virtual Data Base COTS Product

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of information systems science and technology for aerospace command and control and its transition to air, space, and ground systems to meet customer needs in the areas of Global Awareness, Dynamic Planning and Execution, and Global Information Exchange is the focus of this AFRL organization. The directorate's areas of investigation include a broad spectrum of information and fusion, communication, collaborative environment and modeling and simulation, defensive information warfare, and intelligent information systems technologies.